Technical Documentation

Table of Contents

util/db_util.py

db_open()

- creates a connection to the database that MUST BE CLOSED with db_close()

Input Parameter(s)
None

Return Value
connection

- a connection to the database

db_close(connection)

- closes the connection to the database

Input Parameter(s)
connection

- a connection to the database

Return Value
None

db_query(connection, query)

- queries the database for an SQL query that returns results
- returns such results as a list of lists where each nested list represents a tuple

Input Parameter(s)
connection

- a connection to the database

query

- a string containing an SQL query

Return Value
output

- a list of lists

db_execute(connection, query)

- queries the database for an SQL query that does not return results

Input Parameter(s)
connection

- a connection to the database

query

- a string containing an SQL query

Return Value
None

print_output(output)

- prints the output of a list

Input Parameter(s)
output

- a list

Return Value
None

util/db_helper.py

Tasks Table Functions

get_task(taskID)

- gets a particular tuple in the tasks table in the database based on the task ID and returns a list of lists containing that tuple

Input Parameter(s)

taskID

- an int
- represents the task's ID

Return Value

task

- a tuple containing all information for a given tasks

add_task(taskName,description)

- add a tuple to the tasks table in the database

Input Parameter(s)

taskName

- a string
- represents the title/name of the task

description

- a string
- represents the description of the tasks

Return Value

None

modify_task(taskID, taskName=None, description=None, state=None, employeeID=None, timeCreated=None, timeCompleted=None)

- modifies a tuples in the tasks table in the database with given attributes to be changed

Input Parameter(s)

taskID

- in int
- represents the task ID

taskName

- a string
- contains the task name to which the task's task name should be changed

description

- a string
- contains the description to which the task's description should be changed

state

- a string
- contains the state to which the task's state should be changed

employeeID

- an int
- contains the employee's ID to which the task's employeeID should be changed

timeCreated

- a string
- contains the time to which the task's creation time should be changed

timeCompleted

- a string
- contains the time to which the task's completion time should be changed

Return Value
None

remove_task(taskID)

- removes a particular tuple in the tasks table in the database

Input Parameter(s)
taskID

- an int
- represents the task's ID

Return Value
None

get_all_tasks()

- gets all tuples of the tasks table in the database and returns a list of lists containing those tuples

Input Parameter(s)
None

Return Value
tasks

- a list of lists
- contains all tasks
- nested list is a tuple containing all information for a given tasks

get_completed_tasks()

- gets all tuples of the tasks table in the database which have a Complete state and returns a list of lists containing those tuples

Input Parameter(s)
None

Return Value
tasks

- a list of lists
- contains only tasks that are Complete
- nested list is a tuple containing all information for a given tasks

get_noncompleted_tasks()

- gets all tuples in the tasks table in the database which do not have a Complete state and returns a list of lists containing those tuples

Input Parameter(s)
None

Return Value
tasks

- a list of lists

- contains only tasks that are not Complete
- nested list is a tuple containing all information for a given tasks

get_inprogress_tasks()

- gets all tuples of the tasks table in the database which have an In Progress state and returns a list of lists containing those tuples

Input Parameter(s)
None

Return Value
tasks

- a list of lists
- contains only tasks that are In Progress
- nested list is a tuple containing all information for a given tasks

get_incomplete_tasks()

- gets all tuples of the tasks table in the database which have an Incomplete state and returns a list of lists containing those tuples

Input Parameter(s)
None

Return Value
tasks

- a list of lists
- contains only tasks that are Incomplete
- nested list is a tuple containing all information for a given tasks

get_employee_tasks(employeeID)

- gets all tuples of the tasks table in the database which have a In Progress state and have an employee ID matching the input parameter and returns a list of lists containing those tuples

Input Parameter(s)

employeeID

- an int
- represents the employee's ID

Return Value

tasks

- a list of lists
- contains only tasks that are In Progress that have been claimed by the employee with employeeID
- nested list is a tuple containing all information for a given tasks

update_task_state(taskID, new_state, employeeID)

- updates the tuple in the tasks table matching the task ID with a new state and either an employee ID or a time of completion

Input Parameter(s)

taskID

- an int
- represents the task ID

new_state

- a string
- contains the state to which the task's state should be changed

employeeID

- an int
- contains the employee's ID

Return Value

None

Items Table Functions

get_item_info(RFID)

- gets the name and price of a tuple in the items table in the database based on the RFID and returns a list of lists containing that tuple

Input Parameter(s)

RFID

- an int
- represents the item's RFID

Return Values
tasks
- a list of lists
- contains the item with matching RFID
- nested list is a tuple containing the name and price for the item

Employees Table Functions

get_employee(employeeID)

- gets a particular tuple in the employees table in the database based on the employee ID and returns a list of lists containing that tuple

Input Parameter(s)
employeeID
- an int
- represents the employee's ID

Return Value
employee
- a list
- contains the employee's last name at employee[0] and the employee's first name at employee[1]

task_manager/views.py

This is the most relevant file with regards to coding for the Task Manager. The other files are primarily used to coordinate Django implementation. Relevant HTML files are located in task_manager/templates/task_manager/ with names similar to those of the functions. The input parameter `request` is an HTML request for the page.

index(request)

- highly associated with index.html
- obtains all tasks from the database and sends it to index.html to be displayed as a table with a button for each tuple to send the user to a page where the user can modify most attributes of the tuple
- index.html also displays a button to send the user to a page where the user can create a task, which creates an associated tuple in the database

edit(request)

- highly associated with edit.html
- obtains a given task that the user would like to modify and sends it to edit.html to be displayed
- displays text boxes to allow modification of the task name, description, state, employee ID of the employee who claimed the task (if applicable), time of creation, and time of completion
- displays a button to allow submission of the contents in the text boxes, another button to allow the user to delete the task, and a third button to allow the user to return to the index.html page

modify(request)

- highly associated with modify.html
- obtains the task that the user chose to change in edit.html, which is sent to to modify.html to be displayed
- obtains the task updated with the changed that the user wanted enacted, which is also sent to modify.html to be displayed
- displays a button to allow the user to return to the index.html page

create(request)

- highly associated with create.html

- displays the create.html which displays 2 text boxes—one for entering a task name and one for entering a description—and a button to submit the text

created(request)

- associated with index.html
- creates a task with the user entered task name and description, automatically generated task ID and time of creation (which is the current time)
- obtains all tasks including the created task
- displays index.html but with a notice (which can be closed) that the task has been created

delete(request)

- associated with index.html
- deletes the task which user chose to delete in edit.html in the database
- obtains all tasks excluding the deleted task
- displays index.html but with a notice (which can be closed) that the task has been deleted

task_board/views.py

This is the most relevant file with regards to coding for the Task Board. The other files are primarily used to coordinate Django implementation. Relevant HTML files are located in task_board/templates/task_board/ with names similar to those of the functions. The input parameter `request` is an HTML request for the page.

index(request)

- highly associated with index.html
- displays the index.html page with two buttons—one to claim tasks and one to view tasks

claim_tasks(request)

- highly associated with claim_tasks.html
- obtains all incomplete tasks and displays them in a table in claim_tasks.html
- displays button by each tuple to allow the user to claim the task
- displays a button to allow the user to return to the index.html page

view_tasks(request)

- highly associated with view_tasks.html
- obtains and displays all tasks claimed by the given employee (the user)
- displays button by each tuple to allow the user to complete the task
- displays a button to allow the user to return to the index.html page

claim(request)

- highly associated with claim.html
- updates the task claimed by the user, obtains the task, and displays the task in claim.html and notifies the user that the task has been claimed
- displays a button to allow the user to return to the claim_tasks.html page

complete(request)

- highly associated with complete.html
- updates the task completed by the user, obtains the task, and displays the task in claim.html and notifies the user that the task has been claimed
- displays a button to allow the user to return to the view_tasks.html page

check_out/check_out_helper.py

This is a helper program which holds a linked list program with specific methods for the program

check_out/views.py

This is the most relevant file with regards to coding for the Check_out. The other files are primarily used to coordinate Django implementation. Relevant HTML files are located in check_out/templates/check_out/ with names similar to those of the functions. The input parameter `request` is an HTML request for the page.

index(request)

- highly associated with index.html
- displays the index.html page with Add Items, Remove Item, and Finish buttons

finish(request)

- highly associated with finish.html
- displays the finish.html page with Cash and Credit buttons

add_items(request)

- highly associated with add_items.html which is a copy of index.html with added compatibility functionality
- displays the index.html page with Add Items, Remove Item, and Finish buttons